A43

PTO 01-4453

CY=JP DATE=19920831 KIND=A
PN=04-242829

PROGRAM UPDATING DEVICE
[Puroguramu koushin' shori souchi]

Nobuaki Okuzumi, et al.

UNITED STATES PATENT AND TRADEMARK OFFICE
Washington, D.C. October 2001

Translated by: FLS, Inc.

PUBLICATION COUNTRY	(19):	JP
DOCUMENT NUMBER	(11):	040242829
DOCUMENT KIND	(12):	A [PUBLISHED UNEXAMINED APPLICATION]
PUBLICATION DATE	(43):	19920831
APPLICATION NUMBER	(21):	030457
APPLICATION DATE	(22):	19910108
INTERNATIONAL CLASSIFICATION	(51):	G 06 F 9/06
INVENTORS	(72):	OKUZUMI, NOBUAKI; MURAOKA, FUMIKO; YOSHIDA, SADAHIKO
APPLICANT	(71):	FUJITSU, LTD.
TITLE	(54):	PROGRAM UPDATING DEVICE
FOREIGN TITLE	(54A):	PUROGURAMU KOUSHIN' SHORI SOUCHI

(54) [Title of the Invention] /183*

Program Updating Device

[Claim] /184

[Claim 1]

Program updating device characterized by: being provided with an update information generating part [3] and an update reflecting part [5] for a process in which, if there are an old program [1] that is a prescribed source program composed of an array of more than 1 statement, a first program [2a] obtained by making a first update to said old program, and a second program [2b] obtained by making a second update, the first update is reflected in the second program; said update information generating part [3] consisting of

a process in which said old program [1], first program [2a], and second program [2b] are input to it, it generates a first update information table [4a] by subjecting said old program [1] and first program [2a] to an update information generating process in which the first program is used as a new program, it generates a second update information table [4b] by subjecting said old program [1] and second program [2b] to said update information generating process in which the second program is used as said new program, and it generates an old order table and a new order table in which said statements of said old program

---

*Numbers in the margin indicate pagination in the foreign text.

and said new program are assigned an order in accordance with prescribed size correlations between the character strings of said individual statements in said update information generating process,

a process in which said statements of said old program and said statements of said new program are compared in accordance with the orders indicated in said old order table and said new order table and in which the prescribed deletion, insertion, and no-change are discriminated for each of said statements in accordance with the prescribed conditions of said size correlations of both of said statements, and

a process in which, by referring to said discrimination results and comparing said statements of said old program and said new program in the order of the arrangements in the programs, insertion, deletion, shifting, or no-change is determined for each of said statements of said old program and said new program and in which a prescribed update information table expressing said determined conditions is generated; and

said update reflecting part [5] being arranged in a manner such that it collates the deleted, inserted, and shifted items of the first update information table [4a] with a second update information table [4b], generates prescribed control information that reflects the first update in the second program [2b], and outputs it together with the prescribed information of the first and second update information tables.

[Detailed Description of the Invention]

[0001] [Field of Industrial Application]

The present invention pertains to program updating devices that, when two different kinds of modifications are made to a source program of a computer, comprehend the update status and execute a process that reflects one of the updated contents in the other updated program.

[0002] [Prior Art]

For instance, in some cases a so-called package program of a computer system is modified for each user's system and utilized through what is called customization at the source program level.

[0003]

In a case in which a version of this package program is updated with a functional improvement, etc., it is required that the improved function be reflected on the customized programs, as well. As in this example, there are instances in which, if different modifications are made separately to a source program, a new program to which both modifications have been applied without a conflict needs to be generated. For this reason, a method is used in which one new program is further modified in order to reflect one modification in the other.

[0004]

Such modifications are made by storing the sequence numbers assigned to the statements of the original program and by

executing necessary deletion and insertion of the statements required for the program in order to make it easier to check the modification results while maintaining the correspondences with the original program.

[0005]

Therefore, in order to process such modifications automatically, corresponding statements are compared based on the sequence numbers assigned to both source programs, and insertion, deletion, updating, etc. is thereby discriminated and processed.

[0006] [Problems that the Invention is to Solve]

As mentioned above, both source programs must be updated while reserving the sequence numbers of the original program, and this in some cases restricts modifications that would otherwise be possible.

[0007]

Moreover, since the shifting of statements cannot be identified by said discrimination method, the data for the checking is insufficient, and there was a problem in that the number of manhours was increased due to manual checking.

[0008]

The purpose of the present invention is to supply program updating devices capable of identifying the differences between two programs including shifting without depending on sequence

numbers and capable of reflecting the update of one of two separate updates on the other.

[0009] [Means for Solving the Problem]

Figure 1 is a block diagram showing the structure of the present invention. The figure shows the structure of a program updating device, and it is provided with an update information generating part [3] and an update reflecting part [5], which, when there is an old program [1] that is a prescribed source program and that is composed of an arrangement of more than 1 statement, a first program [2a] obtained by making a first update to the old program [1], and a second program [2b] obtained by making a second update, are used for reflecting the first update in the second program [2b].

[0010]

The update information generating part [3] has the old program [1], the first program [2a], and the second program [2b] input to it, and it generates a first update information table [4a] by subjecting the old program [1] and the first program [2a] to an update information generating process in which the first program [2a] is handled as a new program. It also generates a second update information table [4b] by subjecting said old program and the second program [2b] to said update information generating process in which the second program [2b] is handled as said new program.

[0011]

Said update information generating process consists of the following: a process in which, based on said old program [1] and said new program, an old order table and a new order table in which said statements are placed in an order based on prescribed size correlations between the character strings of said statements are generated; a process in which said statements of said old program and said statements of said new program are compared in accordance with the orders indicated in said old order table and said new order table and in which the prescribed /185
deletion, insertion, and no-change of each of said statements is discriminated based on prescribed conditions of said size correlations between said both statements; and a process in which insertion, deletion, shifting, or no-change of each of said statements of said old program and said new program is determined by referring to said discrimination results and by comparing said statements of said old program and said new program in the order of the arrangements in the programs and in which a prescribed update information table expressing said determined conditions is generated.

[0012] The update reflecting part [5] generates prescribed control information that reflects the first update in the second program [2b] by collating the deleted, inserted, and shifted items of the first update information table [4a] and the second update information table [4b], and it outputs this together with the required information of the first and second update

information tables, [2a] and [2b].

[0013] [Operation of the Invention]

When detecting the status of a modification from an old program to a new program, deletion and/or insertion is detected by comparing the results of sorting both programs based on the character strings of the statements in the order of the sorting.

[0014]

A statement for which no change has been discriminated in the above process remains completely unchanged, remains unchanged with the exception of merely being shifted in parallel, or is moved with its positional relationships with others altered. Therefore, shifting can be isolated by comparing statements that were unchanged in the previous process in the order of the program arrangement.

[0015]

In the above manner, based on the update information table that was made concerning the first and second programs, control information that is for reflecting an update performed on the first program in the second program as well is generated automatically.

[0016]

By checking this control information while referring to said update information, etc., the user can update the second program again based on this control information and complete the necessary update of the program after making necessary

corrections, etc.

[0017] [Working Example]

Figure 2 shows the content of the processings of the update information generating part [3] and update reflecting part [5] of the present invention, which are shown in Fig. 1, based on the relationships between the various tables that are generated.

[0018]

The update information generating part [3] first makes an old source table [11], a first source table [12a], and a second source table [12b] based on the old program [1], the first program [2a], and the second program [2b], respectively. Each source table is a table in which the statements of each source program are positioned in the order of the program arrangement and are assigned arrangement order numbers. It has, for example, the structure shown in Fig. 6(a).

[0019]

Next, the statements of each source table are sorted based on their character strings and are assigned an order that, for instance, begins with the smaller ones. Then order tables, such as an old order table [13], a first order table [14a], and a second order table [14b], in which the arrangement order numbers of the source tables are listed in the order of said sorting are generated.

[0020]

Next, by referring to the old order table [13] and the first order table [14a], applicable statements of the two programs are compared. By means of this first matching process, a first association table [15a] that indicates the associations between the same statements from both programs is created. At the same time, when making a modification to the first program [2a], the statement, etc., deleted from the old program [1] is marked in the old source table [11] and the statement, etc. inserted into the first program [2a] is marked in the first source table [12a].

[0021]

Moreover, in the same manner of first matching process in which the old order table [13] and the second order table [14b] are referred to, a deleted and/or inserted statement due to a modification made to the second program [2b] is marked in the old source table [11] and the second source table [12b] while creating a second association table [15b].

[0022]

Each association table is a table that expresses the one-to-one association between the statements of the same character strings of the old and new programs by using arrangement order numbers. It has, for instance, the structure shown in Figure 6(c).

[0023]

Figure 3 is a diagram showing one example of the flow of the

first matching process described above in which each association table is created. By assuming that I and j are item numbers, the character string of a statement in the old source table [11] indicated by the arrangement order number of the ith item of the old order table [13] and the character string of a statement in the new source table (first source table [12a] or second source table [12b]) indicated by the arrangement order number of the jth item of the new order table (first order table [14a] or second order table [14b]) are compared and processed.

[0024]

First, **i** and **j** are initialized to **1** in the processing step [20] of Fig. 3, and then the character strings of the statements of the old and new programs (hereafter referred to as new statement and old statement) defined by **i** and **j** are compared in processing step [21]. If they are equal, the association between the arrangement order numbers of the old and new statements is recorded in the association table (first association table [15a] or second association table [15b]) in processing step [22]. Then, **1** is added to both **i** and **j** in processing step [23].

[0025]

If the old statement is smaller, it is determined that the old statement was deleted, and a flag, **D**, that indicates deletion is recorded in the applicable statement in the old source table [11] in processing step [24]. In processing step [25], the

number corresponding to the arrangement order number on the old statement side of said association table is made to be **0**, and only **i** is increased by **1** in processing step [26].

[0026]

If the old statement is larger, it is determined that a new statement has been inserted, and a flag, **I**, that indicates insertion is recorded in the applicable statement in the new source table in processing step [27]. In processing step [28], the number corresponding to the arrangement order number on the new statement side of said association table is made to be **0**, and only **j** is increased by **1** in processing step [29].

[0027]

After this, it is discerned whether or not one of the order tables was completed in processing step [30], and if not, the above processes are repeated from processing step [21]. In a case in which one has ended, if there are extra items in the old order table [13], D flags are recorded in the old commands indicated by those items, and if there are extra items in the new order table, **I** flags are recorded in the applicable new commands. 186

[0028]

Moreover, as for each of the order tables, [13], [14a], and [14b], **t** is made to be the item number of the order tables and is initialized to the values of the final items of said order tables in processing step [32], as indicated in Fig. 4. Then, the

statements indicated by the arrangement order numbers of the **t**th items of the order tables are compared with the statements indicated by the arrangement order numbers of the (t-1)th items in processing step [33].

[0029]

If both statements are equal, the flag **E** is recorded in the statement indicated by the arrangement order number of the **t**th item in processing step [34]. If they are not equal, nothing is done, **1** is subtracted from **t** in processing step [36], the step is returned to processing step [33], and the above is repeated until the identification result of **t** becomes **1** in processing step [35].

[0030]

By the above processes, **a** flag, **D**, **I**, or **E**, is recorded in the required statements in the old source table [11], the first source table [12a], and the second source table [12b]. A statement without the flag **D** or **I** assigned to it indicates that there is a correspondence between the old and new programs.

[0031]

At this time, the update information generating part [3] compares the statements of the old and new source tables and, as necessary, generates update information tables (a first update information table [4a] and second update information table [4b]) by means of a second matching process in which the association tables are referred to. An update information table consists of,

for example, an item column (corresponding to the processing order), flag column, X-number column, program section column, and Y-number column, as shown in Fig. 6(d).

[0032]

The program section column indicates whether an arrangement order number in the X-number column is a number from the new source table (N) or from the old source table (O) by using **N** or **O**. Y-number values will be described later.

[0033]

Figure 5 is a diagram showing one example of the flow of the second matching process, and it shows a process in which, by using **m** as the arrangement order number of the new source table and **n** as the arrangement order number of the old source table, the character strings of statements in the old source table [11] indicated by the arrangement order numbers, **n**, and the character strings of statements in the new source table indicated by the arrangement order numbers, **m**, are compared and processed in the order of the arrangement order numbers.

[0034]

First, **m** and **n** are initialized to **1** in processing step [40] of Fig. 5. Then in processing step [41], it is discerned whether or not there is a flag, which was set in the above-mentioned process, in the arrangement order number, **m**, or arrangement order number, **n**, of the old and new source table, which are the

processing objects. If there is a flag, the type of the flag is identified in processing step [42].

[0035]

If the flag is **D**, **I**, or **T** (described later), this flag is set as the flag of the next 1 item of the table in processing step [43], and the program section is made to be O if the flag is **D** and **N** if the flag is **I** or **T** in processing step [44]. The arrangement order number (that is to say, **m** or **n** of the specific instance) corresponding to the program section is set as the X number, and the Y number is set in processing step [46].

[0036]

As the Y number, a number that corresponds to the X number in the association tables is set (for instance, if an arrangement order number of the new program is set as the X number, the Y number is the corresponding arrangement order number of the old program in the association tables).

[0037]

If, however, the flag is **I** or **T**, the same value as the Y number of the previous item in this update information table is set. Moreover, if the flag is **D**, the same value as the Y number of the previous item in the table is set if processing the first program, that is to say when generating the first update information table, and the same value as the X number of that item is set if processing the second program. This is for the

convenience of a later-described processing for generating a reflection table.

[0038]

The setting of one item is thus completed. In processing step [53], it is determined whether or not the source table has been processed to the end. If there is an unprocessed item, **1** is added to the arrangement order number ([m], [n], or both) of one that was the object to be registered in the update information table in processing step [54], and the step is returned to processing step [41].

[0039]

If there is no flag in processing step [41], the statement of the arrangement order number, **m**, of the new program and the statement of the arrangement order number, **n**, of the old program are extracted from each source table and the character strings are compared in processing step [47].

[0040]

If the result is not equal, it is considered to be that said statement of the old program has been shifted, and **F** is set as the flag and the shifting destination is indicated in processing step [48]. The statement of the new program that corresponds with the statement of the old program is determined based on the association tables, the flag of said statement of the new source table is set to **T**, and the shifting destination is indicated in

processing step [49].

[0041]

After that, one item of the update information table is completed in the same manner as that above through processing step [43] and thereafter. It should be noted that it is assumed that an **F** flag is processed in the same manner as a **D** flag. If the character strings are equal as a result of comparing the old and new statements in processing step [47], the program section is set to **N** in the next item of the update information table in processing step [50], and then the step is advanced to processing step [45] and one item is completed in the same manner as above. Therefore, the flag column becomes a blank in this case, indicating that the statement is one that is unchanged.

[0042]

If the flag is **E** in the above processing step [42], the old and new statements are compared in processing step [51]. If the character strings are equal, the process is advanced to processing step [50] and the same processing as that above is carried out. If they are not equal, the flag is converted to **D** if the program is the old one or converted to **I** if the program is the new one in processing step [52]. Then, the process is advanced to processing step [43], and the processing is advanced

thereafter as described above.

[0043]

By the above processing, the association table of (c) is created for each source table of the old program and new program of Fig. 6(a), and then the update information table of (d) is obtained.

[0044]

The update information generating part [3] executes the above processes for the first source table [12a] and the old source table [11] and generates the first update information table [4a]. Moreover, it executes them for the second source table [12b] and the old source table [11] and generates the second update information table [4b]. /187

[0045]

Then, the update reflecting part [5] extracts the items with flags assigned to them from the first update information table [4a]. **F** flags are converted to **D**, **T** are converted to **I**, and a differential information table [16] is generated. Based on the differential information table [16] and the second update information table [4b], a reflection table [17] that becomes the data used to reflect the update contents of the first program in the second program is generated. The reflection table has, for

example, the constitution of Fig. 8(d).

[0046]

Figure 7 is a chart showing one example of the flow of the processing for generating the reflection table [17] in which, by using **i** and **j** as item numbers and by advancing the processes by means of comparing the **Y** numbers of the ith item of the second update information table and of the **j**th item of the differential information table, a reflection table [17] is generated based on the content of the applicable items of the second update information table or differential information table.

[0047]

In processing step [60] of Fig. 7, the **i** and **j** values are initialized to 1, and in processing step [61], the Y number of the ith item of the second update information table [4b] and the Y number of the **j**th item of the differential information table are compared.

[0048]

If the result is that the Y number of the **i**th item of the second update information table [4b] is not larger (that is to say is smaller than or equal to), the content of the **i**th item of the second update information table [4b] is set in 1 item of the reflection table in processing step [62]. Next, **1** is added to **i** in processing step [63], and if the table is not completed, the step returns from processing step [64] to processing step [61]

and a comparison is made between the new **i** and the original **j**.

[0049]

If the Y number of the **i**th item of the second update information table [4b] is larger as a result of the comparison in the processing step [61], the **j**th item of the differential information table is set in the next item of the reflection table after changing **D** flags to **C** and **I** to **R** in processing step [65].

[0050]

This item is indicated as an item for which the data needs to be processed by assigning **H** to the program section column in processing step [66]. In processing step [67], X is assigned to the previous program section to indicate that it is the item previous to the item **H**.

[0051]

After inheriting the values of the previous item in the X number and Y number and completing the item in processing step [68], **1** is added to **j** in processing step [69], the step returns to processing step [61] via processing step [64], and the above processes are repeated.

[0052]

Figure 8 shows examples of the results of the above processes. Based on the source table shown in Fig. 8(a) as an

example, each of the update information tables shown in (b) is made. The differential information table shown in (c) is extracted from the first update information table. Based on the second update information table of (b) and the differential information table of (c), the reflection table shown in (d) is obtained through the above-mentioned processes. By referring to this reflection table, the update card shown in Fig. 8(e) as an example can be created and output as one of the update reflecting outputs [6].

[0053]

An update card is made by making the program section of the reflection table correspond to the item **H**. The value of the X-number column of that item indicates the update object of the second program by using the arrangement order number from the second source table, and if the flag is **C**, the applicable statement of the second program is changed (replace command) to a line that only contains a comment and is thus deleted.

[0054]

If the flag is **R**, an update card is made in a manner such that the statement in the first source table that has the value of the Y-number column as the arrangement order number is inserted (insert command) after the statement that is the processing object.

[0055]

In other words, the first command of Fig. 8(e) is generated in correspondence with the second item of the reflection table in order to deal with the flag **C**. This command "-REPLACE 1 * A" is a command for replacing the statement of the arrangement order number "1" with a comment line "* A", and the original statement (A) is left in the comment to indicate the alteration history.

[0056]

Moreover, since the flag of the 6th item in the reflection table is R, "-INSERT 3 A" is generated as indicated in the second command, and a command for inserting the statement "A" (the statement of the arrangement order number "2" of the first program) after the statement of the arrangement order number "3" is provided. Similarly, the third and fourth commands of (e) are obtained in correspondence with the 8th and 11th items of the reflection table.

[0057]

By applying these update cards to the second program indicated in the source table of Fig. 8(a), they become updated as indicated in the "post update" column of the same source table, and the update contents of the first program can be reflected in the second program.

[0058] [Effects of the Invention]

As is evident from the above explanation, in an updating process of a computer source program, differences, including

shifting, between two programs can be identified automatically without depending on sequence numbers, and based on this, control information that reflects one of two separate updates in the other is generated automatically, and therefore, the present invention has significant industrial effects in that program updating can be efficiently processed.

[Brief Descriptions of the Drawings]

[Figure 1] A block diagram showing the constitution of the present invention.

[Figure 2] A diagram for explaining the table generation relationships.

[Figure 3] A flow chart of the first matching process (No.1).

[Figure 4] A flow chart of the first matching process (No.2).

[Figure 5] A flow chart of the second matching process.

[Figure 6] A figure for explaining the examples of the various tables.

[Figure 7] A flow chart of a reflection table generating process.

[Figure 8] A drawing for explaining an example of a processing result.

[Descriptions of Reference Numerals]

[1] = old program

[2a] = first program

[2b] = second program

[3] = update information generating part

[4a] = first update information table

[4b] = second update information table /188

[5] = update reflecting part

[6] = update reflecting output

[11] = old source table

[12a] = first source table

[12b] = second source table

[13] = old order table

[14a] = first order table

[14b] = second order table

[15a] = first association table

[15b] = second association table

[16] = differential information table

[17] = reflection table

[20]~[36], [40]~[54], and [60]~[69] = processing step

[Figure 1] A block diagram showing the constitution of the present invention

1) old program;
2a) first program;
2b) second program;
3) update information generating part;
4a) first update information table;
4b) second update information table;
5) update reflecting part; 6)update reflecting output.

[Figure 2] A diagram for explaining the table generation relationships

1) old program;
2a) first program;

2b) second program;
3) update information generating part;
4a) first update information table;
4b) second update information table;
5) update reflecting part;
6) update reflecting output;
11) old source table;
12a) first source table;
12b) second source table;
13) old order table;
14a) first order table;
14b) second order table;
15a) first association table;
15b) second association table;
16) differential information table;
17) reflection table.

[Figure 4] A flow chart of the first matching process (No.2)
32) t←Final item number of the order table;
33) Statement(t)=Statement(t-1);
34) Flag E for the statement(t) of the source table;
Key: a) Start; b) End.

[Figure 3] A flow chart of the first matching process (No.1)
21) Old statement(I):New statement(j);
22) Old and new arrangement order numbers are associated.;
24) A D flag is set for the new source table.;
25) 0 is set for the new-side value corresponding to the old arrangement number.;
27) An I flag is set for the new source table.;
28) 0 is set for the old-side value corresponding to the new arrangement number.;
30) End?;
31) Flag is supplemented.
Key: a) Start; b) End.

[Figure 5] A flow chart of the second matching process
41) Is there a flag?;
42) The flag is…;
43) A flag is set.;
44) In the program section, N for a flag I or T, and O for a flag D or F.;
45) An arrangement order number corresponding to the program section is set as the X number.;
46) The Y number is made to be a value determined by the X number and the flag.;
47) Command(n)=Command(m);
48) An F flag is set for the old side.;
49) A T flag is set for the new side that corresponds in the

association table.;
50) N for the program section.;
51) Command(n)=Command(m).;
52) The flag is set as D if old side and I if new side.;
54) m or n is updated.;
59) End?;
Key: a) Start; b) End.

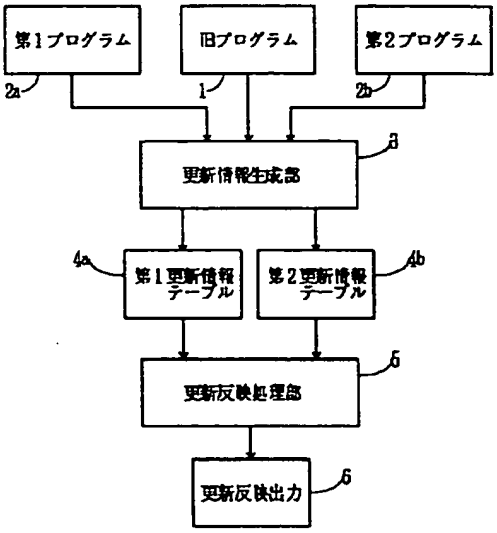
[Figure 6] A figure for explaining the examples of the various tables
Key: a) Flag; b) Arrangement Order Number; c) Old Program; d) New Program.; e) Old Order; f) New Order; g) Number of the Old that Corresponds to the New; h) Number of the New that Corresponds to the Old; i) Item; j) X number; k) Program Section; l) Y number.

[Figure 7] A flow chart of a reflection table generating process
61) Y number(I):Y number(j);
62) The ith item of the second update information table to the reflection table.;
64) End?;
65) The jth item of the differential information table to the reflection table.;
66) The flag is changed to C or R and the program section to H.;
67) The X number and Y number are made to be the values of the previous item.;
68) The program section of the previous item is set to X;
Key: a) Start; b) End.

[Figure 8] A drawing for explaining an example of a processing result
(a) Source Table
(b) First Update Information Table, Second Update Information Table
(c) Differential Information Table
(d) Reflection Table
(e) Update Card
Key: a) Arrangement Order Number; b) Old Program; c) First Program; d) Second program; e) Post Update; f) Item; g) Flag; h) X number; i) Program Section; j) Y number.

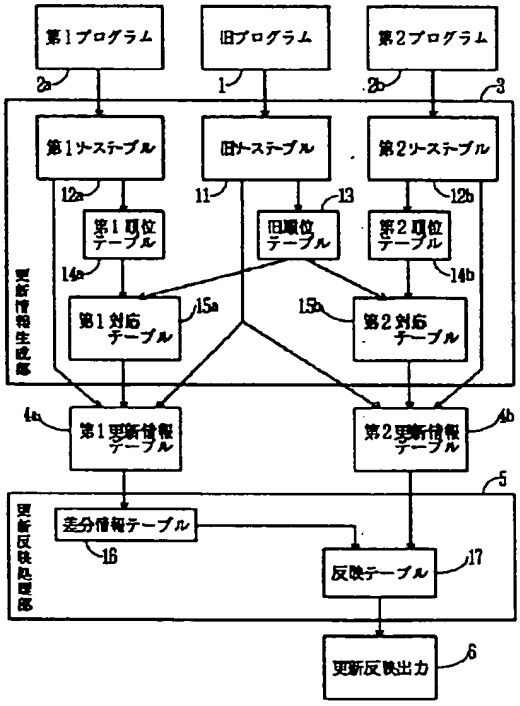
【図1】 [Figure 1]

本発明の構成を示すブロック図



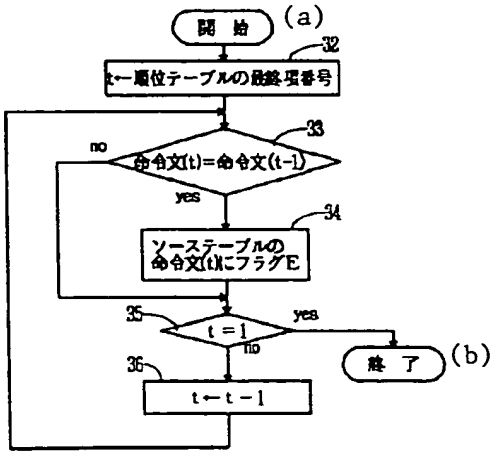
【図2】 [Figure 2]

テーブルの生成関係を説明する図

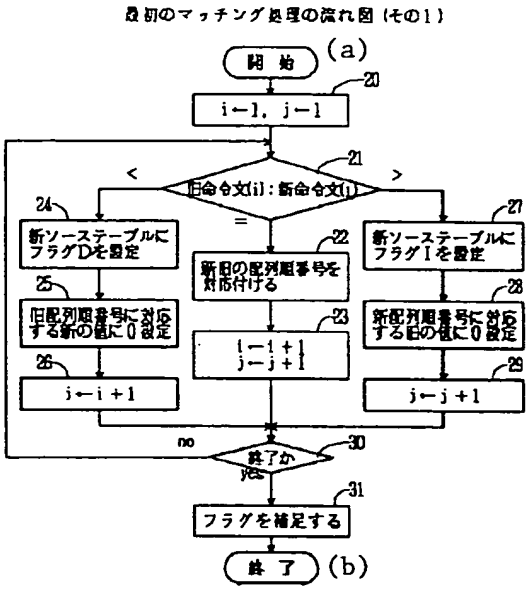


【図4】 [Figure 4]

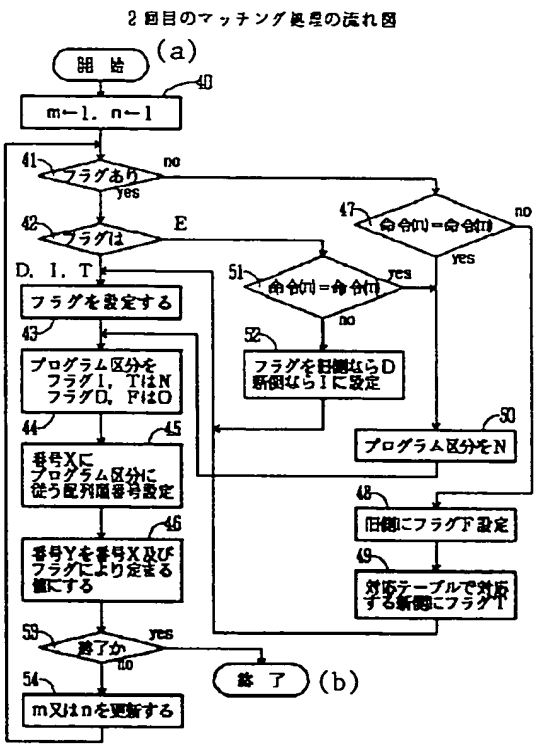
最初のマッチング処理の流れ図（その2）



【図3】[Figure 3]



【図5】[Figure 5]



【図6】[Figure 6]

各種テーブルの例を説明する図

(a)

フラグ	(b) 配列順番号	(c) 旧プログラム
	1	A
	2	B
	3	D
	4	C
	5	E

(d)

(a) フラグ	(b) 配列順番号	(c) 新プログラム
	1	B
	2	A
	3	C
	4	E
	5	F

(e)

旧順位	(b) 配列順番号
1	1
2	2
3	4
4	3
5	5

(f)

新順位	(b) 配列順番号
1	1
2	2
3	3
4	4
5	5

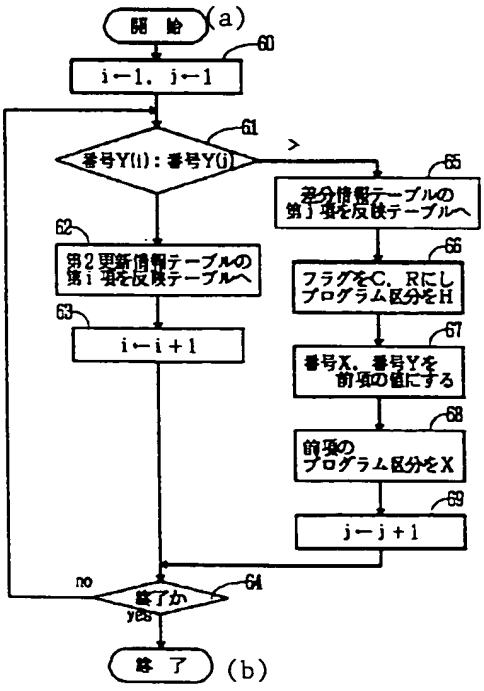
(b) (g) (d) (h)

配列順番号	新に対する旧の番号	旧に対する新の番号
1	2	2
2	1	1
3	4	0
4	5	3
5	0	4

(i) (a) (j) (d) (k) (l)

項	フラグ	番号X	プログラム区分	番号Y
1	F	1	O	1
2		1	N	2
3	T	2	N	3
4	D	4	O	2
5		3	N	4
6		4	N	5
7	I	5	N	5

【図7】[Figure 7]



【図8】

処理結果例の説明図